GENERATING MULTI-SENTENCE ABSTRACTIVE SUMMARIES OF INTERLEAVED TEXTS

Anonymous authors
Paper under double-blind review

ABSTRACT

In multi-participant postings, as in online chat conversations, several conversations or topic threads may take place concurrently. This leads to difficulties for readers reviewing the postings in not only following discussions but also in quickly identifying their essence. A two-step process, disentanglement of interleaved posts followed by summarization of each thread, addresses the issue, but disentanglement errors are propagated to the summarization step, degrading the overall performance. To address this, we propose an end-to-end trainable encoder-decoder network for summarizing interleaved posts. The interleaved posts are encoded hierarchically, i.e., word-to-word (words in a post) followed by post-to-post (posts in a channel). The decoder also generates summaries hierarchically, thread-to-thread (generate thread representations) followed by word-to-word (i.e., generate summary words). Additionally, we propose a hierarchical attention mechanism for interleaved text. Overall, our end-to-end trainable hierarchical framework enhances performance over a sequence to sequence framework by 8-10% on multiple synthetic interleaved texts datasets.

INTRODUCTION

Interleaved texts are becoming more common with new ways of working and new forms of communication, e.g., multi-author entries for activity reports, meeting minutes and social media conversations, such as Slack. Quickly getting a sense of or following the content of different threads in interleaved texts, where posts belonging to different threads occur in one sequence, is often difficult. An example of two threads with multiple posts interleaved to form a sequence is:

Post1-Thread1 → Post1-Thread2 → Post2-Thread1 → Post3-Thread1 → Post2-Thread2 → Post3-Thread2

This intermingling leads to difficulties in not only following discussions but also in retrieving their essence. In conversation disentanglement, interleaved posts are grouped by the thread; e.g., the previous example could be rearranged as:

Post1-Thread1 → Post2-Thread1 → Post3-Thread1
Post1-Thread2 → Post2-Thread2 → Post3-Thread2

In analyzing interleaved texts, Shang et al. (2018) went a step further and proposed summarization of the interleaved texts. They designed an unsupervised two-step system and evaluated the system on meeting texts. In the first step, a conversation disentanglement component disentangles the texts thread-wise. Then, in the second step, a multi-sentence compression component compresses the thread-wise posts to single sentence summaries. However, this system has a major disadvantage, in that the disentanglement obtained through either supervised Jiang et al. (2018) or unsupervised Wang & Oard (2009) methods propagate its errors to the downstream summarization task, and thus, degrades the overall performance.

We aim to tackle this issue of error propagation through an end-to-end trainable encoder-decoder system that takes a variable length input, e.g., interleaved texts, processes it and generates a variable length output, e.g., a multi-sentence summary; see Figure 1 for an illustration. An end-to-end system eliminates the disentanglement component, and thus, the error propagation.
Figure 1: 7 interleaved posts are implicitly disentangled into 3 threads, and single sentence summaries are generated for each thread. Posts are outlined with colors corresponding the color of each summary.

The encoder first performs word-to-word encoding to embed each post, followed by post-to-post encoding to embed the overall content of the posts and represent the discourse structure of the interleaved texts. The decoder has a thread-to-thread decoder to generate a representation for each thread, and the thread representation is given to a word-to-word decoder to generate a summary sentence. We also propose to use hierarchical attention similar to Nallapati et al. (2017), but instead of computing post-level attention at every word, attentions are only computed when decoding new sentences. Further, the attention networks are trained end-to-end.

Despite the availability of a multitude of real-world interleaved texts, a major challenge to train encoder-decoder models is the lack of labels (summaries). As labeling is difficult and expensive Barker et al. (2016); Aker et al. (2016); Verberne et al. (2018), we synthesize two separate corpora. Each is derived by mixing texts and associated summaries from a corpus of documents, where the mixed text has a structure reflective of a multi-party conversation with inter-leaved threads and the summary highlights the information in the threads. We find abstracts and titles of randomized controlled trial (RCT) articles, a PubMed corpus, and also for Stack Exchange posts and titles.

To summarize, our contributions are threefold:

- We propose to combine a hierarchical encoder and decoder to obtain multi-sentence summaries of interleaved texts.
- We propose a novel hierarchical attention mechanism that is integrated with the hierarchical encoder-decoder architecture and which equips the decoder to disentangle the interleaving further.
- We use two synthetic datasets to verify the ideas and show our end-to-end trainable architecture addresses not only the issue of error-propagation observed in competitive methods but also improves the performance.

RELATED WORK

Quite often multi-party conversations, e.g. news comments, social media conversation and activity report, have tens of posts discussing several different matters pertaining to a subject. Ma et al. (2012); Aker et al. (2016); Shang et al. (2018) designed methodologies to summarize posts in order to provide an overview on the discussed matters. They broadly follow the same approach: cluster the posts and then extract a summary from each cluster.

There are two kinds of summarization: abstractive and extractive. In abstractive summarization, the model utilizes a corpus level vocabulary and generates novel sentences as the summary, while extractive models extract or rearrange the source words as the summary. Abstractive models based on neural sequence-to-sequence (seq2seq) Rush et al. (2015) proved to generate summaries with higher ROUGE scores than the feature-based abstractive models. Integration of attention into seq2seq Bah-
There are many possible patterns of organization of the information in texts, e.g., chronological pattern. News articles have an inverted pyramid pattern, i.e., core information in the lead sentences and the extra information in later sentences. A seq2seq model is appropriate for summarization of a news article as it encodes and decodes sequentially. However, in interleaved texts, related information maybe separated; thus a seq2seq model may be competent but not sufficient.

Li et al. (2015) proposed an encoder-decoder (auto-encoder) model that utilizes a hierarchy of networks: word-to-word followed by sentence-to-sentence. Their model is better at capturing the underlying structure than a vanilla sequential encoder-decoder model (seq2seq). Krause et al. (2017), Jing et al. (2018) showed multi-sentence captioning of an image through a decoder based on a hierarchical Recurrent Neural Network (RNN), topic-to-topic followed by word-to-word, is better than seq2seq.

These works suggest a hierarchical encoder, with word-to-word encoding followed by post-to-post, will better recognize the dispersed information in interleaved texts. Similarly, a hierarchical decoder, thread-to-thread followed by word-to-word, will intrinsically disentangle the posts, and therefore, generate more appropriate summaries.

Nallapati et al. (2016) devised a hierarchical attention mechanism for a seq2seq model, where two levels of attention distributions over the source, i.e., sentence and word, are computed at every step of the word decoding. Based on the sentence attentions, the word attentions are rescaled. Hsu et al. (2018) slightly simplified this mechanism and computed the sentence attention only at the first step. Our hierarchical attention is more intuitive and computes new sentence attentions for every new summary sentence, and unlike Hsu et al. (2018), is trained end-to-end.

Figure 2: Hierarchical encoder-decoder architecture.

**Problem Statement**

We aim to design a system that when given a sequence of posts, \( C = \langle P_1, \ldots, P_{|C|} \rangle \), produces a sequence of summaries, \( T = \langle S_1, \ldots, S_{|T|} \rangle \). For simplicity and clarity, unless otherwise noted, we will use lowercase italics for variables, uppercase italics for sequences, lowercase bold for vectors and uppercase bold for matrices.

Figure 2 illustrates the proposed hierarchical encoder-decoder framework. In the framework, first, a low-level, word-to-word encoder converts a sequence of words in a post, \( P_j \), to a sequence of representations, \( H_j = (h_{j0}^{w2w}, \ldots, h_{jP_j}^{w2w}) \). Subsequently, a top-level, post-to-post encoder converts those representations, \( (H_0, \ldots, H_{|C|}) \), to a sequence of top-level post representations \( (h_{0}^{p2p}, \ldots, h_{|C|}^{p2p}) \).

These encoded representations are then passed to the decoder, which utilizes a top-level, thread-to-thread, decoder to disentangle them into a sequence of thread representations \( (h_{1}^{t2t}, \ldots, h_{|T|}^{t2t}) \). Finally, a low-level, word-to-word, decoder takes a feed-forward mapped thread representation \( h_{i}^{t2t} \) and generates a sequence of summary words \( (w_{i1}, \ldots, w_{i|S_i|}) \).

The maximum number of posts in a sequence of interleaved texts is denoted by \( n \) and threads by \( m \). We limit the number of words in posts and summaries to fixed lengths by either truncating or
padding, and denote them by \( p \) and \( q \) respectively. The hidden states of the encoder and decoder have dimensionality \( l \).

**ENCODER**

The hierarchical encoder used in Figure 2 is based on \([\text{Nallapati et al. (2017)}]\), where word-to-word and post-to-post encoders are bi-directional LSTMs. The word-to-word Bi-LSTM encoder \((E_{w2w})\) runs over word embeddings of any post, \( P_j \), and generates a set of hidden representations \((h_{0}^{E_{w2w}}, \ldots, h_{P_j}^{E_{w2w}})\); see left middle section in Figure 2. The average pooled value of the word-to-word representations of post \( P_j \) (\( \frac{1}{p} \sum_{k=0}^{p} h_{j+k}^{E_{w2w}} \)) are input to the post-to-post Bi-LSTM encoder \((E_{l2l})\), which then generates a set of representations \((h_{0}^{E_{l2l}}, \ldots, h_{n}^{E_{l2l}})\) corresponding to those posts. In summary, the encoding process results into a tensor of word-to-word representations, \( W \), with dimension \( n \times p \times 2l \) and a matrix of post-to-post representations, \( P \), with dimension \( n \times 2l \) for a given channel, \( C \).

**DECODER**

Our hierarchical decoder (see Figure 2) is based on \([\text{Li et al. (2015)}]\), where both thread-to-thread and word-to-word decoders are uni-directional LSTMs.

The initial state \( h_{0}^{D_{l2t}} \) of the thread-to-thread LSTM decoder \((f_{D_{l2t}})\) is set with a feedforward-mapped representation of an average pooled post representations \((c' = \frac{1}{n} \sum_{n=0}^{n} h_{k}^{E_{l2l}})\). At each step \( i \) of the decoder, a sequence of attention weights, \( (\beta_{i,0}, \ldots, \beta_{i,n \times p}) \), corresponding to the set of encoded word representations, \( (h_{0}^{E_{w2w}}, \ldots, h_{n}^{E_{w2w}}) \), are computed utilizing the previous state, \( h_{i-1}^{D_{l2t}} \). We will elaborate the attention computation in the next section.

![Figure 3: A step in the thread-to-thread decoder.](image)

A weighted representation of the words (crossed blue circle) is then computed: \( \sum_{j=1}^{n} \beta_{ij} \cdot W_{ij} \), and used as one input to LSTM \( f_{D_{l2t}} \). Additionally, we use the last hidden state \( h_{n}^{D_{l2t}} \) of the word-to-word decoder LSTM \((D_{w2w})\) of the previously generated summary sentence as the second input to compute the next state of thread-to-thread decoder, i.e., \( h_{i}^{D_{l2t}} \). The motivation is to provide information about the previous sentence.

The current state \( h_{i}^{D_{l2t}} \) is passed through a single layer feedforward network and a distribution over STOP=1 and CONTINUE=0 is computed:

\[
p_{i}^{STOP} = \sigma(g(h_{i}^{D_{l2t}})) \tag{1}
\]

where \( g \) is a feedforward network. In Figure 3 the process is depicted by a red circle. The thread-to-thread decoder keeps decoding until \( p_{i}^{STOP} \) is greater than 0.5.

Additionally, the current state \( h_{i}^{D_{l2t}} \) is passed through another single layer feedforward network \( k \) followed by \( \text{tanh} \) activation to compute the thread representation \( s_{i} = \text{tanh}(k(h_{i}^{D_{l2t}})) \).

Given a thread representation \( s_{i} \), the word-to-word decoder generates a summary for the thread. Our word-to-word decoder is based on \([\text{Bahdanau et al. (2014)}]\). It is a unidirectional attentional LSTM \((f_{D_{w2w}})\); see the right-hand side of Figure 2. We refer to \([\text{Bahdanau et al. (2014)}]\) for further details.
**Hierarchical Attention**

Our novel dynamic hierarchical attention is based on the idea of making attention distributions interdependent. We draw inspiration from See et al. (2017) wherein a model learns to merge attention distribution over source tokens with the attention distribution over vocabulary tokens using switch probabilities, and thereby, determines which tokens to copy and which to pull from vocabulary in the process of generating a summary.

In our hierarchical attention, the higher level attention (corresponding to source tokens), \( \beta \), that is computed while obtaining a thread representation, \( s \), scales the lower level attention (also corresponding to source tokens), \( \alpha \), that is computed while generating a word, \( y \), of a summary, \( S \); see yellow rectangles in Figure 4.

**Figure 4:** Hierarchical attention mechanism for a hierarchical encoder.

At step \( i \) of thread-to-thread decoding, we compute elements of \( \beta \) as below.

\[
\beta_{ik} = \sigma(\text{attn}^\beta(h_{Dt}^{2t-1}, a_k)) \quad k \in \{1, \ldots, n \times p\}
\]  

where \( a_k \) is computed as in Eq. 3 and \( \text{attn}^\beta \) is a feedforward network that aligns the current thread decoder state \( h_{Dt}^{2t-1} \) with all \( n \times p \) number of \( a_k \) vectors.

Further, at step \( i \) of thread-to-thread decoding, we also compute elements of post-level attention, i.e., \( \gamma \) as below.

\[
\gamma_{iq} = \sigma(\text{attn}^\beta(h_{Dt}^{2t-1}, P_q)) \quad q \in \{1, \ldots, n\}
\]  

Then, we use \( \gamma \) to rescale high-level attention, \( \beta \) as below.

\[
\hat{\beta}_{ik} = \hat{\beta}_{ik} \times \gamma_{il} \quad k := j \times l, \quad j \in \{1, \ldots, p\}, \quad l \in \{1, \ldots, n\}
\]  

At step \( t \) of word-to-word decoding, we compute elements of \( \alpha_t \) as below.

\[
\alpha_{tk} = \frac{\exp(e_{tk})}{\sum_{k=1}^{n \times p} \exp(e_{tk})}
\]  

where \( a_k \) is same as in Eq. 3 and \( \text{attn}^\alpha \) is a feedforward network that aligns the current word decoder state \( h_{Dw}^{2w-1} \) with all \( n \times p \) number of \( a_k \) vectors.

Finally, we use rescaled high-level word attentions, \( \hat{\beta} \), for rescaling a low level attention:

\[
\hat{\alpha}_{jk} = \frac{\hat{\beta}_k \times \exp(e_{jk})}{\sum_{k=1}^{n \times p} \hat{\beta}_k \times \exp(e_{jk})}
\]  

Table 1 compares different forms of hierarchical attention.
Table 1: Comparing different forms of hierarchical attention. HierDcode=Hierarchical Decoding

<table>
<thead>
<tr>
<th>Model</th>
<th>End2End</th>
<th>HierDcode(many β)</th>
<th>reuse</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nallapati et al.</td>
<td>yes</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>Hsu et al.</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>Li, Luong, and Jurafsky</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>Ours</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
</tbody>
</table>

TRAINING OBJECTIVE

We train our hierarchical encoder-decoder network similarly to an attentive seq2seq model [Bahdanau et al. (2014)], but with an additional weighted sum of sigmoid cross-entropy loss on stopping distribution; see Eq. 1. Given a summary, \( Y_i = \langle w_i0, \ldots, w_iq \rangle \), our word-to-word decoder generates a target \( \hat{Y} = \langle y_i0, \ldots, y_iq \rangle \), with words from a same vocabulary \( U \). We train our model end-to-end by minimizing the objective given in Eq. 9.

\[
\sum_{i=1}^{m} \sum_{j=1}^{q} \log p_\theta(y_{ij} | w_i \cdot <j, W>) + \lambda \sum_{i=1}^{m} y_i^{STOP} \log(p_i^{STOP})
\]  

DATASET

Obtaining labeled training data for conversation summarization is challenging. The available ones are either extractive (Verberne et al. (2018)) or too small (Barker et al. (2016); Anguera et al. (2012)) to train a neural model. To get around this issue and verify the proposed architecture, we synthesized a dataset by utilizing a corpus of conventional texts for which summaries are available. The PubMed corpus contains a type of article, i.e., randomized controlled trials (RCT), where sentences in the abstract are structured into sections and the title of the article is an abstractive summary of the information from these sections (Dernoncourt & Lee (2017)). Further, Stack Exchange is a question-and-answer site for several diverse fields topics. In this dataset, every question (with multiple sentences) has a title, and zero, one or many answers. The title forms the gist or abstract of the question. Thus, a random interleaving of sentences from a few pubmed abstracts or Stack Exchange questions roughly resembles interleaved texts, and correspondingly interleaving of titles resembles its multi-sentence summary.

The algorithm that we devised for creating synthetic interleaved texts is defined in detail in the Appendix. Given a set of abstracts and titles and a range for \( I \), a set of concatenated interleaved texts and summaries are returned. The number of abstracts to include in the interleaved texts is given as a range (from \( a \) to \( b \)) and the number of sentences per abstract to include is given as a second range (from \( m \) to \( n \)).

We vary INTERLEAVE parameters as below, and create three different corpora for experiments:

- Easy: \( a=2, b=2, m=5 \) and \( n=5 \)
- Medium: \( a=2, b=3, m=2 \) and \( n=5 \)
- Hard: \( a=2, b=5, m=2 \) and \( n=5 \)

Table 2 shows an example of a data instance in the Hard Interleaved RCT corpus.

EXPERIMENTS

Evaluation Metrics: we report ROUGE-1 (unigram), ROUGE-2 (bigram), and ROUGE-L (longest-common substring) as the quantitative evaluation of the models.

Parameters: Pubmed Medium and Hard corpora train, test and validation have approximately 290k, 6k and 1.5k instances, respectively. In case of Stack-Exchange, the medium corpus train has 180k and Hard has 210k instances, while the both corpora have test and validation of approximately 5k and 4k instances, respectively. Remaining hyper-parameters is described in detail in the Appendix.
A study was conducted to evaluate the influence of supervised fitness walking in patients with osteoarthritis of the knee, a randomized, controlled trial.

A 9-week randomized, controlled trial...

Nine endurance-trained athletes participated in a randomized...

We examined the effects of intensity of training on ratings of perceived exertion.

Subjects were recruited as sedentary controls or were randomly assigned to the light or moderate intensity training groups. The at least group trained at a velocity of 5.5 and the greater than...

Data were obtained on 47 of 51 intervention patients and 45 control patients.

Caffeine in sport: influence of endurance exercise on the urinary caffeine concentration.

Supervised fitness walking in patients with osteoarthritis of the knee: a randomized, controlled trial.

This effect of training intensity on ratings of perceived exertion.

Table 2: The top rows contain interleaving of 3 articles with 2 to 5 sentences, bottom contains their interleaved titles.

**BASELINES**

We reimplemented the Bahdanau et al. (2014) seq2seq model, and evaluated it on a standard task of news summarization. We use a popular, CNN/DailyMail, dataset [Napoles et al. (2012)], with approximately 300k training examples for the purpose. We fetched the test set from See et al. (2017) and report the results on it. The results are compared with the seq2seq methods of Nallapati et al. (2016) and See et al. (2017). As our aim for this experiment is to demonstrate the strength of the reimplemented, we set the parameters to produce comparable results in less computation time (2 days compared to others weeks). We achieve Rouge-1 and Rouge-2 of 29.58 and 9.36 respectively. Nallapati et al. reported 32.49 and 11.84 and See et al. reported 31.33 and 11.81 for the same. This indicates that our implementation is competitive to the latter tuned models.

We then take the implemented seq2seq model and train and test it on the Easy corpus. We also ran Shang et al. (2018)'s two-step system on the test set of the Easy Interleaved Pubmed corpus. As the Shang et al. (2018) system is unsupervised, it doesn’t need training. Additionally, we also utilized Shang et al. (2018)'s clustering component to first cluster the interleaved texts of the corpus, and then the disentangled corpus is used to train the seq2seq model. We refer to the latter as cluster→seq2seq. The performance comparison of Shang et al. (2018) and the two seq2seq models are shown in Table 3. Clearly, seq2seq performs better than Shang et al. (2018), the reason being a seq2seq model trained on a sufficiently large dataset is better at summarization than the unsupervised sentence compression (extractive) method. The lower performance of cluster→seq2seq in comparison to seq2seq shows not only that a disentanglement component is unnecessary but also illustrates the error propagation of disentanglement to summarization.

<table>
<thead>
<tr>
<th>Model</th>
<th>Rouge-1</th>
<th>Rouge-2</th>
<th>Rouge-L</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shang et al. (2018)</td>
<td>44.38</td>
<td>19.47</td>
<td>35.20</td>
</tr>
<tr>
<td>seq2seq</td>
<td>30.37</td>
<td>10.77</td>
<td>20.04</td>
</tr>
<tr>
<td>cluster→seq2seq</td>
<td>42.93</td>
<td>18.76</td>
<td>30.68</td>
</tr>
</tbody>
</table>

Table 3: Rouge F1-Scores for seq2seq models on the Pubmed Easy Corpus.

**SEQ2SEQ VS. HIER2HIER MODELS**

We then compare the proposed hierarchical approach against the seq-to-seq approach in summarizing the interleaved texts by experimenting on the Medium and Hard corpora.

Table 4 shows the experimental results on several corpora: clearly, an increase in the complexity of interleaving scantly impacts the performances. Though, in case of Pubmed, seq2seq deprecates by 1-2 points while hier2hier remains nearly same with increase of complexity. In Stack Exchange corpora, as Hard corpora has more training data (≈30k) seq2seq improves slightly. However, a noticeable improvement is observed on changing the decoder to hierarchical, i.e., 1.5-3 Rouge points in Pubmed and 2-4.5 points in Stack Exchange depending on the interleaving complexity.

To understand the impact of hierarchy on the hier2hier model, we perform an ablation study and use the Hard Pubmed corpus for the experiments, and Table 5 shows the results. Clearly, adding hierarchical decoding already provides a boost in the performance. Hierarchical encoding also adds

---

Under review as a conference paper at ICLR 2020
Table 4: Rouge Recall-Scores of models on the Medium and Hard Corpus.

<table>
<thead>
<tr>
<th>Corpus Difficulty</th>
<th>Model</th>
<th>PubmedRCT Rouge-1</th>
<th>2 Rouge</th>
<th>3 Rouge-L</th>
<th>Stack Exchange Rouge-1</th>
<th>2 Rouge</th>
<th>3 Rouge-L</th>
</tr>
</thead>
<tbody>
<tr>
<td>Medium</td>
<td>seq2seq</td>
<td>30.67</td>
<td>11.71</td>
<td>23.80</td>
<td>18.78</td>
<td>03.52</td>
<td>14.73</td>
</tr>
<tr>
<td></td>
<td>hier2hier</td>
<td><strong>32.78</strong></td>
<td><strong>12.36</strong></td>
<td><strong>25.33</strong></td>
<td><strong>24.34</strong></td>
<td><strong>05.07</strong></td>
<td><strong>18.63</strong></td>
</tr>
<tr>
<td>Hard</td>
<td>seq2seq</td>
<td>29.07</td>
<td>10.96</td>
<td>21.76</td>
<td>20.21</td>
<td>04.03</td>
<td>14.93</td>
</tr>
<tr>
<td></td>
<td>hier2hier</td>
<td><strong>33.36</strong></td>
<td><strong>12.69</strong></td>
<td><strong>24.72</strong></td>
<td><strong>24.96</strong></td>
<td><strong>05.56</strong></td>
<td><strong>17.95</strong></td>
</tr>
</tbody>
</table>

some improvements to the performance; however, the enhancement attained in training and inference speed by the hierarchical encoding is much more valuable. We will discuss it in depth later.

Table 5: Rouge Recall-Scores of models on the Hard Pubmed Corpus.

<table>
<thead>
<tr>
<th>Model</th>
<th>Hard Pubmed Corpus Rouge-1</th>
<th>2 Rouge</th>
<th>3 Rouge-L</th>
</tr>
</thead>
<tbody>
<tr>
<td>seq2seq</td>
<td>29.07</td>
<td>10.96</td>
<td>21.76</td>
</tr>
<tr>
<td>seq2hier</td>
<td>32.92</td>
<td>11.87</td>
<td>24.43</td>
</tr>
<tr>
<td>hier2seq</td>
<td>31.86</td>
<td>11.9</td>
<td>23.57</td>
</tr>
<tr>
<td>hier2hier</td>
<td><strong>33.36</strong></td>
<td><strong>12.69</strong></td>
<td><strong>24.72</strong></td>
</tr>
</tbody>
</table>

Table 6: Rouge Recall-Scores of models on the Stack Exchange Medium and Hard Corpus.

<table>
<thead>
<tr>
<th>Model</th>
<th>Medium Corpus Rouge-1</th>
<th>2 Rouge</th>
<th>3 Rouge-L</th>
<th>Hard Corpus Rouge-1</th>
<th>2 Rouge</th>
<th>3 Rouge-L</th>
</tr>
</thead>
<tbody>
<tr>
<td>seq2seq</td>
<td>19.67</td>
<td>03.88</td>
<td>15.37</td>
<td>19.62</td>
<td>03.71</td>
<td>14.90</td>
</tr>
<tr>
<td>hier2hier</td>
<td><strong>23.97</strong></td>
<td><strong>05.63</strong></td>
<td><strong>18.75</strong></td>
<td><strong>24.14</strong></td>
<td><strong>05.00</strong></td>
<td><strong>17.25</strong></td>
</tr>
</tbody>
</table>

Importantly, hier2hier converges much earlier than the seq2seq and also reaches a lower training loss (Figure 5 in Appendix). hier2hier takes ≈1.5 days for training on a Tesla V100 GPU, while seq2seq takes ≈4.5 days. Therefore, the hier2hier model not only achieves greater accuracy but also reduces training and inference time.

Hierarchical Attention

The contribution of the post-level and high-level attentions in a hierarchical decoder is two-fold: computing the thread representation and re-scaling the word-level attentions. To understand the impact of hierarchical attention on the hier2hier model, we perform an ablation study of post-level attentions ($\gamma$) and high-level attentions ($\beta$), using the Hard corpus for the experiments.

Table 7 shows the performance comparison. Clearly, $\gamma$ attention improves the performance (0.5-1) to hierarchical decoding but not a lot. The high-level attention, i.e., $\beta$ is very important as without it the model performances is noticeably reduced (Rouge values decrease from 2-3). Additionally, we also include Li, Luong and Jurafsky, 2015 post-level attention technique in the comparison, as it is the closest hierarchical attention to ours (see Table 1). Though, (Li, Luong and Jurafsky) utilize a softmax obtained $\beta$ to compute thread representation, and in contrast to us, they do not reuse $\beta$ for re-scaling $\alpha$, and thereby, closest to hier2hier($+\gamma-\beta$). Evidently, re-utilization gives hier2hier($+\gamma-\beta$) an edge over (Li, Luong and Jurafsky, 2015). Lastly, removing both the $\gamma$ and $\beta$...
Table 7: Rouge Recall-Scores of models on the Hard Pubmed Corpus.

Table 8: Interleaved sentences of 3 articles, and corresponding ground-truth and hier2hier generated summaries. The top 2 sentences that were attended ($\gamma$) for the generation are in left. Additionally, top words ($\beta$) attended for the generation are colored accordingly.

**DISCUSSION**

Occurrence of interleaved texts is common; however, readers often don’t have the patience to wade through them. Surveys have shown that summarized contents are easy to consume, and thus, many organizations have begun displaying extractive summaries, content visualization or the combination alongside the interleaved texts. Table 8 shows an output of our hierarchical abstractive system, in which interleaved texts are in the top, and ground-truth and generated summaries in the bottom. Table 8 also shows the top two post indexes attended to by the post-level attention ($\gamma$) while generating those summaries, and they coincide with relevant posts. Similarly, the top 10 indexes (words) of the high-level attention ($\beta$) is directly visualized in the table through the color coding matching the generation. The system not only manages to disentangle the interleaved texts but also to generate appropriate abstractive summaries. Meanwhile, $\beta$ provides explainability of the output.

As interleaving in the table includes abstracts from the same category, e.g. Physical Activities, the interleaving is complex and approximates the real-world conversations. Despite that, the end-to-end hierarchical system tackles the task to a large extent. The next, future stage in this research is transfer learning of the hierarchical system to more types of conversations. We find a Stack-Exchange Hard corpora ($a=6$, $b=10$, $m=2$ and $n=3$) trained along with the popular meeting AMI corpus (McCowan et al. (2005)) already reaches competitive results (Rouge-1=25.9, Rouge-2=03.10) against SOA abstractive system on AMI (e.g. Rouge-1=21.7 and Rouge-2=02.5 of TextRank (Mihalcea & Tarau (2004)), Rouge-1=27.9, Rouge-2=04.10 of FUSION (Mehdad et al. (2013))). Additionally, we aim to modify hier2hier to include some of the recent additions of seq2seq models, e.g., see et al 2017 copy mechanism.

1 http://resources.trustyou.com/c/wp-present-travel-review-content?x=0MFT5U
CONCLUSION

We presented an end-to-end trainable hierarchical encoder-decoder architecture which implicitly disentangles interleaved texts and generates a multi-sentence abstractive summary covering the text threads. Furthermore, the architecture addresses the error propagation issue that occurs in the two-step architectures. Our proposed novel hierarchical attention further boosts both disentanglement and summary generation.

REFERENCES


APPENDIX

In Algorithm. [1] INTERLEAVE takes a set of concatenated abstracts and titles, \( C = \langle A_1; T_1, \ldots, A_{|C|}; T_{|C|} \rangle \), minimum, \( a \), and maximum, \( b \), number of abstracts to interleave, and minimum, \( m \), and maximum, \( n \), number of sentences in a source, and then returns a set of concatenated interleaved texts and summaries. WINDOW takes a sequence of texts, \( X \), and returns a window iterator of size \( |X| - w + 1 \), where \( w \) and \( t \) are window size and sliding step respectively. window reuses elements of \( X \), and therefore, enlarges the corpus size. Notations \( \mathcal{U} \) refers to a uniform sampling, \([\cdot]\) to array indexing, and \( \text{REVERSE} \) to reversing an array.

Algorithm 1 Interleaving Algorithm

1: `procedure INTERLEAVE(C, a, b, m, n)`
2: \( \hat{C}, Z \leftarrow \text{WINDOW}(C, w = b, t = 1), \text{Array()} \)
3: while \( \hat{C} \neq \emptyset \) do
4: \( C', A', T', S \leftarrow \hat{C}.\text{NEXT}(), \text{Array()}, \text{Array}(), \{\} \)
5: \( r \sim \mathcal{U}(a, b) \)
6: for \( j = 1 \) to \( r \) do \( \triangleright \) Selection
7: \( A, T \leftarrow \hat{C}[j] \)
8: \( T'.\text{ADD}(T) \)
9: \( q \sim \mathcal{U}(m, n) \)
10: \( A'.\text{ADD}(A[1:q]) \)
11: \( S \leftarrow S \cup \{j \times q\} \)
12: \( \hat{A}, \hat{T} \leftarrow \text{Array}(), \text{Array}() \)
13: for \( 1 \) to \( |S| \) do \( \triangleright \) Interleaving
14: \( k \leftarrow \mathcal{U}(S) \)
15: \( S \leftarrow S \setminus k \)
16: \( \hat{I} \leftarrow \text{REVERSE}(A'[k]).\text{POP}() \)
17: \( \hat{A}.\text{ADD}(\hat{I}) \)
18: \( J \leftarrow \hat{T}[k] \)
19: if \( J \notin \hat{T} \) then:
20: \( \hat{T}.\text{ADD}(J) \)
21: \( Z.\text{ADD}(\hat{A}; \hat{T}) \)
22: `return Z`

PARAMETERS

For the word-to-word encoder, the steps are limited to 20, while the steps in the word-to-word decoder are limited to 14. The steps in the post-to-post encoder and thread-to-thread decoder depend on the corpus type, e.g., Medium has 15 steps in post-to-post and 3 steps in thread-to-thread. In seq2seq experiments, the source is flattened, and therefore, the number of steps in the source is limited to 300. We initialized all weights, including word embeddings, with a random normal distribution with mean 0 and standard deviation 0.1. The embedding vectors are of dimension 80. The vocabulary size is limited to 5000 and 15000 for Pubmed and Stack Exchange corpora respectively. All hidden states of the encoder and decoder in the models are set to dimension 100. We pad short sequences with a special token, \( \langle \text{PAD} \rangle \). We use Adam [Kingma & Ba 2014] with an initial learning rate of .0001 and batch size of 64 for training. Texts are lowercased and numbers are replaced by the special symbol \%.
Figure 5: A comparison of running average training loss between seq2seq (pink) and hier2hier (gray) for Stack Exchange Hard corpus.